Multi-style Generative Network for Real-time Transfer

Hang Zhang[1,2] Kristin Dana[2]

[1]Amazon AI [2]Rutgers University
hzaws@amazon.com, kdana@ece.rutgers.edu

Abstract. Despite the rapid progress in style transfer, existing approaches using feed-forward generative network for multi-style or arbitrary-style transfer are usually compromised of image quality and model flexibility. We find it is fundamentally difficult to achieve comprehensive style modeling using 1-dimensional style embedding. Motivated by this, we introduce CoMatch Layer that learns to match the second order feature statistics with the target styles. With the CoMatch Layer, we build a Multi-style Generative Network (MSG-Net), which achieves real-time performance. In addition, we employ an specific strategy of upsampled convolution which avoids checkerboard artifacts caused by fractionally-strided convolution. Our method has achieved superior image quality comparing to state-of-the-art approaches. The proposed MSG-Net as a general approach for real-time style transfer is compatible with most existing techniques including content-style interpolation, color-preserving, spatial control and brush stroke size control. MSG-Net is the first to achieve real-time brush-size control in a purely feed-forward manner for style transfer. Our implementations and pre-trained models for Torch, PyTorch and MXNet frameworks will be publicly available[1].

1 Introduction

Style transfer can be approached as reconstructing or synthesizing texture based on the target image semantic content [1]. Many pioneering works have achieved success in classic texture synthesis starting with methods that resample pixels [2–5] or match multi-scale feature statistics [6–8]. These methods employ traditional image pyramids obtained by handcrafted multi-scale linear filter banks [9, 10] and perform texture synthesis by matching the feature statistics to the target style. In recent years, the concepts of texture synthesis and style transfer have been revisited within the context of deep learning. Gatys *et al.* [11] shows that using feature correlations (*i.e.* Gram Matrix) of convolutional neural nets (CNN) successfully captures the image styles. This framework has brought a surge of interest in texture synthesis and style transfer using iterative optimization [1, 11, 12] or training feed-forward networks [13–16]. Recent work extends style flexibility using feed-forward networks and achieves multistyle or arbitrary style transfer [17–20]. These approaches typically encode image styles into 1-dimensional space, *i.e.* tuning the featuremap mean and variance (bias and scale) for different styles. However, the comprehensive appearance of image style is fundamentally difficult to represent in 1D embedding space. Figure 3 shows style transfer results

[1] links can be found at http://hangzhang.org/

Fig. 1: Examples of transferred images and the corresponding styles using the proposed MSG-Net.

using the optimization-based approach [12] and we can see Gram matrix representation produces more appealing image quality comparing to mean and variance of CNN featuremap.

In addition to the image quality, concerns about the flexibility of current feed-forward generative models have been raised in Jing *et al.* [21], and they point out that no generative methods can adjust the brush stroke size in real-time. Feeding the generative network with high-resolution content image usually results in unsatisfying images as shown in Figure 6. The generative network as a fully convolutional network (FCN) can accept arbitrary input image sizes. Resizing the style image changes the relative brush size and the multistyle generative network matching the image style at run-time should naturally enable brush-size control by changing the input style image size. *What limits the current generative model from being aware of the brush size?* The 1D style embedding (featuremap mean and variance) fundamentally limits the potential of exploring finer behavior for style representations. Therefore, a 2D method is desired for finer representation of image styles.

As the **first contribution** of the paper, we introduce an *CoMatch Layer* which embeds style with a 2D representation and learns to match the second-order feature statistics (Gram Matrix) of the style targets inherently during the training. The CoMatch Layer is differentiable and end-to-end learnable with existing generative network architectures without additional supervision. The proposed CoMatch Layer enables multi-style generation from a single feed-forward network.

The **second contribution** of this paper is building *Multi-style Generative Network (MSG-Net)* with the proposed CoMatch Layer and a novel *Upsample Convolution*. The MSG-Net as a feed-forward network runs in real-time after training. Generative networks typically have a decoder part recovering the image details from downsampled representations. Learning fractionally-strided convolution [22] typically brings checkerboard artifacts. For improving the image quality, we employ a strategy we call *upsampled convolution*, which successfully avoids the checkerboard artifacts by applying an

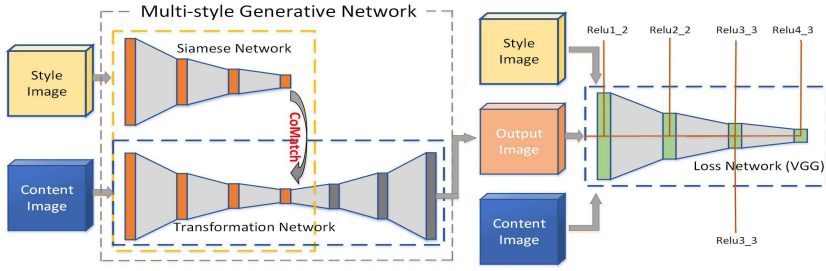

Fig. 2: An overview of MSG-Net, Multi-style Generative Network. The transformation network explicitly matches the features statistics of the style targets captured by a Siamese network using the proposed CoMatch Layer (introduced in Section 3). A pre-trained loss network provides the supervision of MSG-Net learning by minimizing the content and style differences with the targets as discussed in Section 4.2.

(a) Input (b) Mean & Var (c) Gram Matrix

Fig. 3: Comparing 1D and 2D style representation using an optimization-based approach [12]. (a) Input image and style. (b) Style transfer result minimizing difference of CNN featuremap mean and variance. (c) Style transfer result minimizing the difference in Gram matrix representation.

integer stride convolution and outputs an upsampled featuremap (details in Section 4.1). In addition, we extend the Bottleneck architecture [23] to an *Upsampling Residual Block*, which reduces computational complexity without losing style versatility by preserving larger number of channels. Passing identity all the way through the generative network enables the network to extend deeper and converge faster. The experimental results show that MSG-Net has achieved superior image fidelity and test speed compared to previous work. We also study the scalability of the model by extending 100-style MSG-Net to 1K styles using a larger model size and longer training time, and we observe no obvious quality differences. In addition, MSG-Net as a general multi-style strategy is compatible to most existing techniques and progress in style transfer, such as content style trade-off and interpolation [17], spatial control, color preserving and brush-size control [24, 25].

To our knowledge, MSG-Net is the first to achieve real-time brush-size control in a purely feed-forward manner for multistyle transfer.

1.1 Related Work

Relation to Pyramid Matching. Early methods for texture synthesis were developed using multi-scale image pyramids [4, 6–8]. The discovery in these earlier methods was that realistic texture images could be synthesized from manipulating a white noise image so that its feature statistics were matched with the target at each pyramid level. Our approach is inspired by classic methods, which match feature statistics within the feed-forward network, but it leverages the advantages of deep learning networks while placing the computational costs into the training process (feed-forward vs. optimization-based).

Relation to Fusion Layers. Our proposed CoMatch Layer is a kind of fusion layer that takes two inputs (content and style representations). Current work in fusion layers with CNNs include feature map concatenation and element-wise sum [26–28]. However, these approaches are not directly applicable, since there is no separation of style from content. For style transfer, the generated images should not carry semantic information of the style target nor styles of the content image.

Relation to Generative Adversarial Training. The Generative Adversarial Network (GAN) [29], which jointly trains an adversarial generator and discriminator simultaneously, has catalyzed a surge of interest in the study of image generation [26, 27, 30–39]. Recent work on image-to-image GAN [26] adopts a conditional GAN to provide a general solution for some image-to-image generation problems. For those problems, it was previously hard to define a loss function. However, the style transfer problem cannot be tackled using the conditional GAN framework, due to missing ground-truth image pairs. Instead, we follow the work [13, 14] to adopt a discriminator/loss network that minimizes the perceptual difference of synthesized images with content and style targets and provides the supervision of the generative network learning. The initial idea of employing Gram Matrix to trigger style synthesis is inspired by a recent work [30] that suggests using an encoder instead of random vector in GAN framework.

Recent Work in Multiple or Arbitrary Style Transfer. Recent/concurrent work explores multiple or arbitrary style transfer [17, 17, 18, 20]. A style swap layer is proposed in [20], but gets lower quality and slower speed (compared to existing feed-forward approaches). An adaptive instance normalization is introduced in [18] to match the mean and variance of the feature maps with the style target. Instead, our CoMatch Layer matches the second order statistics of Gram Matrices for the feature maps. We also explore the scalability of our approach in the Experiment Section 5.

2 Content and Style Representation

CNNs pre-trained on a very large dataset such as ImageNet can be regarded as descriptive representations of image statistics containing both semantic content and style information. Gatys *et al.* [12] provides explicit representations that independently model the image content and style from CNNs, which we briefly describe in this section for completeness.

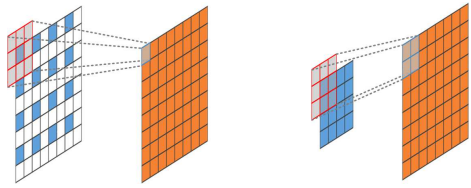

Fig. 4: Left: fractionally-strided convolution. Right: Upsampled convolution, which reduces the checkerboard artifacts by applying an integer stride convolution and outputting an upsampled featuremaps.

The semantic content of the image can be represented as the activations of the descriptive network at i-th scale $\mathcal{F}^i(x) \in \mathbb{R}^{C_i \times H_i \times W_i}$ with a given the input image x, where the C_i, H_i and W_i are the number of feature map channels, feature map height and width. The texture or style of the image can be represented as the distribution of the features using Gram Matrix $\mathcal{G}(\mathcal{F}^i(x)) \in \mathbb{R}^{C_i \times C_i}$ given by

$$\mathcal{G}\left(\mathcal{F}^i(x)\right) = \sum_{h=1}^{H_i} \sum_{w=1}^{W_i} \mathcal{F}^i_{h,w}(x) \mathcal{F}^i_{h,w}(x)^T. \tag{1}$$

The Gram Matrix is orderless and describes the feature distributions. For zero-centered data, the Gram Matrix is the same as the covariance matrix scaled by the number of elements $C_i \times H_i \times W_i$. It can be calculated efficiently by first reshaping the feature map $\Phi\left(\mathcal{F}^i(x)\right) \in \mathbb{R}^{C_i \times (H_i W_i)}$, where $\Phi()$ is a reshaping operation. Then the Gram Matrix can be written as $\mathcal{G}\left(\mathcal{F}^i(x)\right) = \Phi\left(\mathcal{F}^i(x)\right) \Phi\left(\mathcal{F}^i(x)\right)^T$.

3 CoMatch Layer

In this section, we introduce *CoMatch Layer*, which explicitly matches second order feature statistics based on the given styles. For a given content target x_c and a style target x_s, the content and style representations at the i-th scale using the descriptive network can be written as $\mathcal{F}^i(x_c)$ and $\mathcal{G}(\mathcal{F}^i(x_s))$, respectively. A direct solution $\hat{\mathcal{Y}}^i$ is desirable which preserves the semantic content of input image and matches the target style feature statistics:

$$\begin{aligned} \hat{\mathcal{Y}}^i = & \underset{\mathcal{Y}^i}{\operatorname{argmin}} \{ \|\mathcal{Y}^i - \mathcal{F}^i(x_c)\|_F^2 \\ & + \alpha \|\mathcal{G}(\mathcal{Y}^i) - \mathcal{G}\left(\mathcal{F}^i(x_s)\right)\|_F^2 \}. \end{aligned} \tag{2}$$

where α is a trade-off parameter that balancing the contribution of the content and style targets.

The minimization of the above problem is solvable by using an iterative approach, but it is infeasible to achieve it in real-time or make the model differentiable. However, we can still approximate the solution and put the computational burden to the training

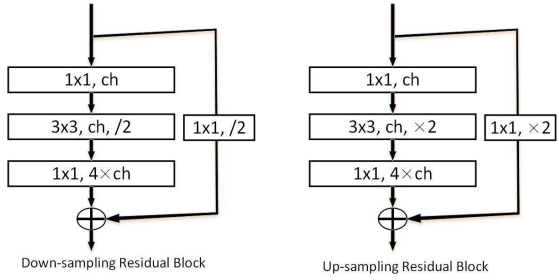

Fig. 5: We extend the original down-sampling residual architecture (left) to an up-sampling version (right). We use a 1×1 fractionally-strided convolution as a shortcut and adopt reflectance padding.

(a) input (b) MSG-Net (ours) (c) baseline

Fig. 6: Comparing Brush-size control. a) High-resolution input image and dense styles. b) Style transfer results using MSG-Net with brush-size control. c) Standard generative network [14] without brush-size control. See also Figure 8

stage. We introduce an approximation which tunes the feature map based on the target style:

$$\hat{\mathcal{Y}}^i = \Phi^{-1}\left[\Phi\left(\mathcal{F}^i(x_c)\right)^T W \mathcal{G}\left(\mathcal{F}^i(x_s)\right)\right]^T, \tag{3}$$

where $W \in \mathbb{R}^{C_i \times C_i}$ is a learnable weight matrix and $\Phi()$ is a reshaping operation to match the dimension, so that $\Phi\left(\mathcal{F}^i(x_c)\right) \in \mathbb{R}^{C_i \times (H_i W_i)}$. For intuition on the functionality of W, suppose $W = \mathcal{G}\left(\mathcal{F}^i(x_s)\right)^{-1}$, then the first term in Equation 2 (content term) is minimized. Now let $W = \Phi\left(\mathcal{F}^i(x_c)\right)^{-T} \mathcal{L}(\mathcal{F}^i(x_s))^{-1}$, where $\mathcal{L}\left(\mathcal{F}^i(x_s)\right)$ is obtained by the Cholesky Decomposition of $\mathcal{G}\left(\mathcal{F}^i(x_s)\right) = \mathcal{L}\left(\mathcal{F}^i(x_s)\right)\mathcal{L}\left(\mathcal{F}^i(x_s)\right)^T$, then the second term of Equation 2 (style term) is minimized. We let W be learned directly from the loss function to dynamically balance the trade-off. The CoMatch Layer

is differentiable and can be inserted in the existing generative network and directly learned from the loss function without any additional supervision.

4 Multi-style Generative Network

4.1 Network Architecture

Prior feed-forward based single-style transfer work learns a generator network that takes only the content image as the input and outputs the transferred image, i.e. the generator network can be expressed as $G(x_c)$, which implicitly learns the feature statistics of the style image from the loss function. We introduce a *Multi-style Generative Network* which takes both content and style target as inputs. i.e. $G(x_c, x_s)$. The proposed network explicitly matches the feature statistics of the style targets at runtime.

As part of the Generator Network, we adopt a Siamese network sharing weights with the encoder part of transformation network, which captures the feature statistics of the style image x_s at different scales, and outputs the Gram Matrices $\{\mathcal{G}(\mathcal{F}^i(x_s))\}(i = 1, ...K)$ where K is the total number of scales. Then a transformation network takes the content image x_c and matches the feature statistics of the style image at multiple scales with CoMatch Layers.

Upsampled Convolution. Standard CNN for image-to-image tasks typically adopts an encoder-decoder framework, because it is efficient to put heavy operations (style switching) in smaller featuremaps and also important to keep a larger receptive field for preserving semantic coherence. The decoder part learns a fractionally-strided convolution to recover the detail information from downsampled featuremaps. However, the fractionally strided convolution [22] typically introduces checkerboard artifacts [40]. Prior work suggests using upsampling followed by convolution to replace the standard fractionally-strided convolution [40]. However, this strategy will decrease the receptive field and it is inefficient to apply convolution on an upsampled area. For this, we use *upsampled convolution*, which has an integer stride, and outputs upsampled featuremaps. For an upsampling factor of 2, the upsampled convolution will produce a 2×2 outputs for each convolutional window as visualized in Figure 4. Comparing to fractionally-strided convolution, this method has the same computation complexity and 4 times parameters. This strategy successfully avoid upsampling artifacts in the network decoder.

Upsample Residual Block. Deep residual learning has achieved great success in visual recognition [23,41]. Residual block architecture plays an important role by reducing the computational complexity without losing diversity by preserving the large number of feature map channels. We extend the original architecture with an upsampling version as shown in Figure 5 (right), which has a fractionally-strided convolution [22] as the shortcut and adopts reflectance padding to avoid artifacts of the generative process. This upsampling residual architecture allows us to pass identity all the way through the network, so that the network converges faster and extends deeper.

Fig. 7: Content and style trade-off and interpolation.

Brush Stroke Size Control. Feeding the generative model with high-resolution image usually results in unsatisfying style transfer outputs, as shown in Figure 6 (c). Controlling brush stroke size can be achieved using optimization-based approach [25]. Resizing the style image changes the brush-size, and feed-forward generative model matches the feature statistics at runtime should naturally achieve brush stoke size control. However, prior work is mainly limited by the 1D style embedding, because this finer style behavior cannot be captured using merely featuremap mean and variance. With MSG-Net, the CoMatch Layer matching the second order statistics elegantly solves the brush-size control. During training, we train the network with different style image sizes to learn from different brush stroke sizes. After training, the brush stroke size can be an option to the user by changing style input image size. Note that the MSG-Net can accept different input sizes for style and content images. Example results are shown in Figure 8.

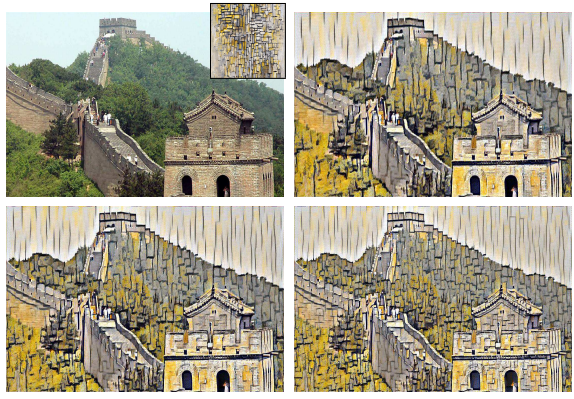

Fig. 8: Brush-size control using MSG-Net. Top left: High-resolution input image and dense style. Others: Style transfer results using MSG-Net with brush-size control.

Other Details. We only use in-network down-sample (convolutional) and up-sample (upsampled convolution) in the transformation network. We use reflectance padding to avoid artifacts at the border. Instance normalization [16] and ReLU are used after weight layers (convolution, fractionally-strided convolution and the CoMatch Layer), which improves the generated image quality and is robust to the image contrast changes.

4.2 Network Learning

Style transfer is an open problem, since there is no gold-standard ground-truth to follow. We follow previous work to minimize a weighted combination of the style and content differences of the generator network outputs and the targets for a given pre-trained loss network $\mathcal{F}$ [13, 14]. Let the generator network be denoted by $G(x_c, x_s)$ parameterized by weights W_G. Learning proceeds by sampling content images $x_c \sim X_c$ and style images $x_s \sim X_s$ and then adjusting the parameters W_G of the generator $G(x_c, x_s)$ in order to minimize the loss:

$$\begin{aligned}\hat{W}_G = \underset{W_G}{\operatorname{argmin}}\, E_{x_c,x_s}\{ & \\ & \lambda_c \|\mathcal{F}^c\left(G(x_c, x_s)\right) - \mathcal{F}^c(x_c)\|_F^2 \\ & + \lambda_s \sum_{i=1}^{K} \|\mathcal{G}\left(\mathcal{F}^i(G(x_c, x_s))\right) - \mathcal{G}(\mathcal{F}^i(x_s))\|_F^2 \\ & + \lambda_{TV}\ell_{TV}\left(G(x_c, x_s)\right)\},\end{aligned} \tag{4}$$

where λ_c and λ_s are the balancing weights for content and style losses. We consider image content at scale c and image style at scales $i \in \{1, ..K\}$. $\ell_{TV}()$ is the total variation regularization as used prior work for encouraging the smoothness of the generated images [14, 42, 43].

Fig. 9: The tradeoff between style-flexibility and output-image quality is challenging for generative models. Our approach enables multi-style transfer and has minimal difference in quality compared to the optimization-based Gatys approach [12].

5 Experimental Results

5.1 Style Transfer

Baselines. We use the implementation of the work of Gatys *et al.* [12] as a gold standard baseline for style transfer approach (technical details will be included in the supplementary material). We also compare our approach with state-of-the-art multistyle or arbitrary style transfer methods, including patch-based approach [20] and 1D style embedding [17, 18]. The implementations from original authors are used in this experiments.

Method Details. We adapt 16-layer VGG network [44] pre-trained on ImageNet as the loss network in Equation 4, because the network features learned from a diverse set of images are likely to be generic and informative. We consider the style representation at 4 different scales using the layers ReLU1_2, ReLU2_2, ReLU3_3 and ReLU4_3, and use the content representation at the layer ReLU2_2. The Microsoft COCO dataset [45] is used as the content image image set X_c, which has around 80,000 natural images. We collect 100 style images, choosing from previous work in style transfer. Additionally 900 real paintings are selected from the open-source artistic dataset wikiart.org [46]

	Model-size	Speed (256)	Speed (512)
Gatys *et al.* [12]	N/A	0.07	0.02
Johnson et al. [14]	6.7MB	91.7	26.3
Dumoulin *et al.* [17]	6.8MB	88.3	24.7
Chen et al. [20]	574MB	5.84	0.31
Huang et al. [18]	28.1MB	37.0	10.2
MSG-Net-100 (ours)	9.6MB	92.7	29.2
MSG-Net-1K (ours)	40.3MB	47.2	14.3

Table 1: Comparing model size on disk and inference/test speed fps (frames/sec) of images with the size of 256×256 and 512×512 on a NVIDIA Titan Xp GPU average over 50 samples. MSG-Net-100 and MSG-Net-1K have 2.3M and 8.9M parameters respectively.

as additional style images for training MSG-Net-1K. We follow the work [13, 14] and adopt Adam [47] to train the network with a learning rate of 1×10^{-3}. We use the loss function as described in Equation 4 with the balancing weights $\lambda_c = 1, \lambda_s = 5, \lambda_{TV} = 1 \times 10^{-6}$ for content, style and total regularization. We resize the content images $x_c \sim X_c$ to 256×256 and learn the network with a batch size of 4 for 80,000 iterations. We iteratively update the style image x_s every iteration with size from $\{256, 512, 768\}$ for runtime brush-size control. After training, the MSG-Net as a fully convolutional network [22] can accept arbitrary input image size. For comparing the style transfer approaches, we use the same content image size, by resizing the image to 512 along the long side. Our implementations are based on Torch [48], PyTorch [49] and MXNet [50]. It takes roughly 8 hours for training MSG-Net-100 model on a Titan Xp GPU.

Model Size and Speed Analysis For mobile applications or cloud services, the model size and test speed are crucial. We compare the model size and inference/test speed of style transfer approaches in Table 1. Our proposed MSG-Net-100 has a comparable model size and speed with single style network [13, 14]. The MSG-Net is faster than Arbitrary Style Transfer work [18], because of using a learned compact encoder instead of pre-trained VGG network.

Qualitative Comparison Our proposed MSG-Net achieves superior performance comparing to state-of-the-art generative network approaches as shown in Figure 9. One may argue that the arbitrary style work has better scalability/capacity [18, 20]. The style flexibility and image quality are always hard trade-off for generative model, and we particularly focus on the image quality in this work. More examples of the transfered images using MSG-Net are shown in Figure 12.

Model Scalability. Prior work using 1D style embedding has achieved success in the scalability of style transfer towards the goal of arbitrary style transfer [18]. To test the scalability of MSG-Net, we augment the style set to 1K images, by adding 900 extra images from the wikiart.org [46]. We also build a larger model MSG-Net-1K with larger model capacity by increasing the width/channels of the model at mid stage (64×64) by a factor of 2, resulting in 8.9M parameters. We also increase the training iterations by

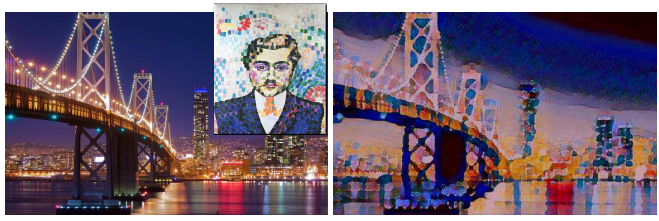

Fig. 10: Color control using MSG-Net, (left) content and style images, (right) color-preserved transfer result.

Fig. 11: Spatial control using MSG-Net. Left: input image, middle: foreground and background styles, right: style transfer result. (Input image and segmentation mask from Shen *et al.* [51, 52].)

4 times (320K) and follow the same training procedure as MSG-Net-100. We observe no quality degradation when increasing the number of styles (examples shown in the supplementary material).

5.2 Runtime Manipulation

MSG-Net as a general approach for real-time style transfer is compatible with existing recent progress for both feed-forward and optimization methods, including but not limited to: content-style trade-off and interpolation (Figure 7), color-preserving transfer (Figure 10), spatial manipulation (Figure 11) and brush stroke size control (Figure 6&8). For style interpolation, we use an affine interpolation of our style embedding following the prior work [17, 18]. For color pre-serving, we match the color of style image with the content image as Gatys *et. al.* [24]. Brush-size control has been discussed in the Section 4.1. We use the segmentation mask provided by Shen *et al.* [51] for spatial control. The source code and technical detail of runtime manipulation will be included in our PyTorch implementation.

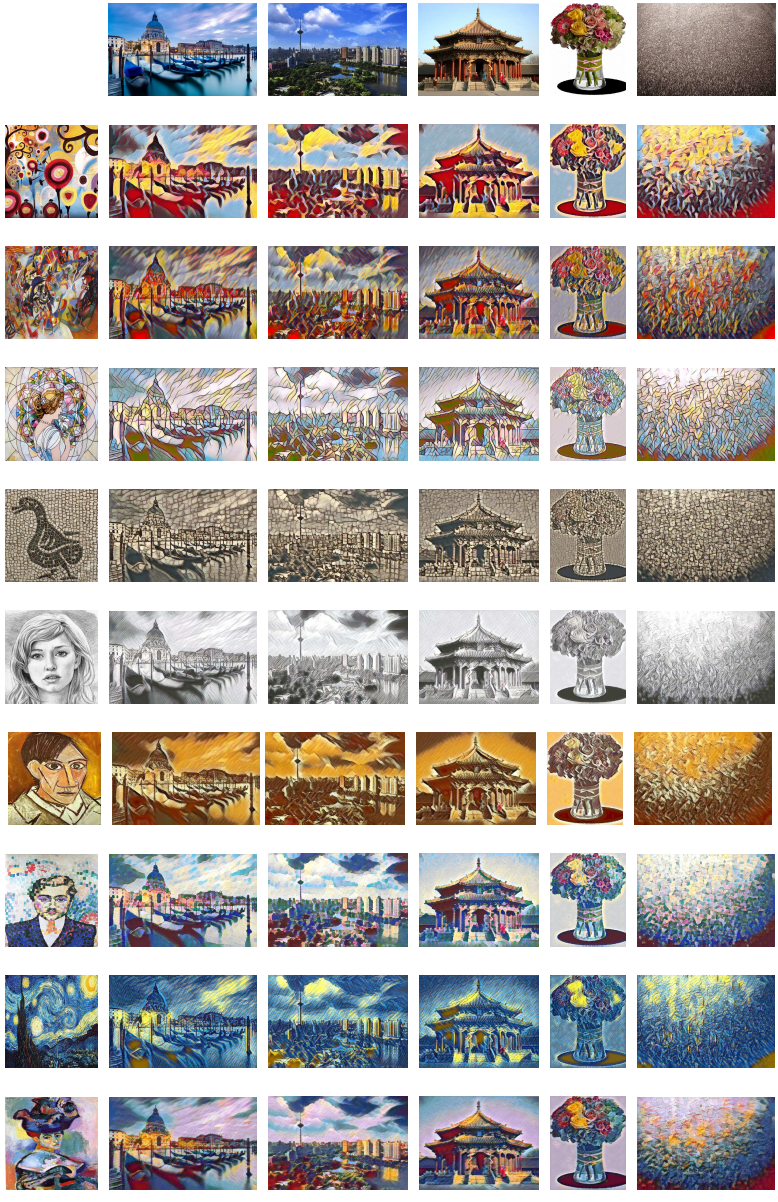

Fig. 12: Diverse images that are generated using a single MSG-Net-100 (2.3M parameters). First row shows the input content images and the other rows are generated images with different style targets (first column).

6 Conclusion and Discussion

To improve the quality and flexibility of generative models in style transfer, we introduce a novel CoMatch Layer that learns to match the second order statistics as image style representation. Multi-style Generative Network has achieved superior image quality comparing to state-of-the-art approaches. In addition, the proposed MSG-Net is compatible with most existing techniques and recent progress of stye transfer including style interpolation, color-preserving and spatial control. Moreover, MSG-Net first enables real-time brush-size control in a fully feed-forward manor. The compact MSG-Net-100 model has only 2.3M parameters and runs at more than 90 fps (frame/sec) on NVIDIA Titan Xp for the input image of size 256×256 and at 15 fps on a laptop GPU (GTX 750M-2GB).

References

1. Li, C., Wand, M.: Combining markov random fields and convolutional neural networks for image synthesis. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)
2. Efros, A.A., Leung, T.K.: Texture synthesis by non-parametric sampling. In: Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on. Volume 2., IEEE (1999) 1033–1038
3. Efros, A.A., Freeman, W.T.: Image quilting for texture synthesis and transfer. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, ACM (2001) 341–346
4. Wei, L.Y., Levoy, M.: Fast texture synthesis using tree-structured vector quantization. In: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (2000) 479–488
5. Kwatra, V., Schödl, A., Essa, I., Turk, G., Bobick, A.: Graphcut textures: image and video synthesis using graph cuts. In: ACM Transactions on Graphics (ToG). Volume 22., ACM (2003) 277–286
6. De Bonet, J.S.: Multiresolution sampling procedure for analysis and synthesis of texture images. In: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co. (1997) 361–368
7. Heeger, D.J., Bergen, J.R.: Pyramid-based texture analysis/synthesis. In: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM (1995) 229–238
8. Portilla, J., Simoncelli, E.P.: A parametric texture model based on joint statistics of complex wavelet coefficients. International journal of computer vision **40**(1) (2000) 49–70
9. Simoncelli, E.P., Freeman, W.T.: The steerable pyramid: A flexible architecture for multi-scale derivative computation. In: Image Processing, 1995. Proceedings., International Conference on. Volume 3., IEEE (1995) 444–447
10. Burt, P., Adelson, E.: The laplacian pyramid as a compact image code. IEEE Transactions on communications **31**(4) (1983) 532–540
11. Gatys, L., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: Advances in Neural Information Processing Systems. (2015) 262–270
12. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 2414–2423

13. Ulyanov, D., Lebedev, V., Vedaldi, A., Lempitsky, V.: Texture networks: Feed-forward synthesis of textures and stylized images. In: Int. Conf. on Machine Learning (ICML). (2016)
14. Johnson, J., Alahi, A., Fei-Fei, L.: Perceptual losses for real-time style transfer and super-resolution. In: European Conference on Computer Vision. (2016)
15. Li, C., Wand, M.: Precomputed real-time texture synthesis with markovian generative adversarial networks. In: European Conference on Computer Vision, Springer (2016) 702–716
16. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. arXiv preprint arXiv:1701.02096 (2017)
17. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. (2016)
18. Huang, X., Belongie, S.: Arbitrary style transfer in real-time with adaptive instance normalization. arXiv preprint arXiv:1703.06868 (2017)
19. Chen, D., Yuan, L., Liao, J., Yu, N., Hua, G.: Stylebank: An explicit representation for neural image style transfer. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)
20. Chen, T.Q., Schmidt, M.: Fast patch-based style transfer of arbitrary style. arXiv preprint arXiv:1612.04337 (2016)
21. Jing, Y., Yang, Y., Feng, Z., Ye, J., Song, M.: Neural style transfer: A review. arXiv preprint arXiv:1705.04058 (2017)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 3431–3440
23. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 770–778
24. Gatys, L.A., Bethge, M., Hertzmann, A., Shechtman, E.: Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897 (2016)
25. Gatys, L.A., Ecker, A.S., Bethge, M., Hertzmann, A., Shechtman, E.: Controlling perceptual factors in neural style transfer. arXiv preprint arXiv:1611.07865 (2016)
26. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. arXiv preprint arXiv:1611.07004 (2016)
27. Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., Wang, X., Metaxas, D.: Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint arXiv:1612.03242 (2016)
28. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2016) 1933–1941
29. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Advances in neural information processing systems. (2014) 2672–2680
30. Che, T., Li, Y., Jacob, A.P., Bengio, Y., Li, W.: Mode regularized generative adversarial networks. arXiv preprint arXiv:1612.02136 (2016)
31. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015)
32. Huang, X., Li, Y., Poursaeed, O., Hopcroft, J., Belongie, S.: Stacked generative adversarial networks. arXiv (2016)
33. Sindagi, V.A., Patel, V.M.: Generating high-quality crowd density maps using contextual pyramid cnns. In: 2017 IEEE International Conference on Computer Vision (ICCV), IEEE (2017)
34. Zhang, H., Sindagi, V., Patel, V.M.: Image de-raining using a conditional generative adversarial network. arXiv preprint arXiv:1701.05957 (2017)

35. Xian, W., Sangkloy, P., Lu, J., Fang, C., Yu, F., Hays, J.: Texturegan: Controlling deep image synthesis with texture patches. arXiv preprint (2018)
36. Zhang, Z., Xie, Y., Yang, L.: Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
37. Zhang, H., Patel, V.M.: Density-aware single image de-raining using a multi-stream dense network. arXiv preprint arXiv:1802.07412 (2018)
38. Zhang, H., Patel, V.M.: Densely connected pyramid dehazing network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
39. Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., He, X.: Attngan: Fine-grained text to image generation with attentional generative adversarial networks. arXiv preprint (2017)
40. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016)
41. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision, Springer (2016) 630–645
42. Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 5188–5196
43. Zhang, H., Yang, J., Zhang, Y., Huang, T.S.: Non-local kernel regression for image and video restoration. In: European Conference on Computer Vision, Springer (2010) 566–579
44. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
45. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European Conference on Computer Vision, Springer (2014) 740–755
46. Duck, S.Y.: Painter by numbers. https://www.kaggle.com/c/painter-by-numbers (2016)
47. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
48. Collobert, R., Kavukcuoglu, K., Farabet, C.: Torch7: A matlab-like environment for machine learning. In: BigLearn, NIPS Workshop. Number EPFL-CONF-192376 (2011)
49. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. (2017)
50. Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z.: Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274 (2015)
51. Shen, X., Hertzmann, A., Jia, J., Paris, S., Price, B., Shechtman, E., Sachs, I.: Automatic portrait segmentation for image stylization. In: Computer Graphics Forum. Volume 35., Wiley Online Library (2016) 93–102
52. Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A.: Context encoding for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2018)